Apurv Sarode 2019130054

Jaiwin Shah 2019130057

Satyam Rai 2019130051

TE COMPS

Batch C

AI/ML Lab

MINI-Project

Problem Statement: Weather forecasting with machine learning.

Theory:

What is K-means Clustering?

K-Means Clustering is an Unsupervised Learning algorithm, which groups the unlabeled dataset into different clusters. Here K defines the number of pre-defined clusters that need to be created in the process, as if K=2, there will be two clusters, and for K=3, there will be three clusters, and so on. It allows us to cluster the data into different groups and a convenient way to discover the categories of groups in the unlabeled dataset on its own without the need for any training.

It is a centroid-based algorithm, where each cluster is associated with a centroid. The main aim of this algorithm is to minimize the sum of distances between the data point and their corresponding clusters.

The algorithm takes the unlabeled dataset as input, divides the dataset into k-number of clusters, and repeats the process until it does not find the best clusters. The value of k should be predetermined in this algorithm.

The working of the K-Means algorithm is explained in the below steps:

Step-1: Select the number K to decide the number of clusters.

Step-2: Select random K points or centroids. (It can be other from the input dataset).

Step-3: Assign each data point to their closest centroid, which will form the predefined K clusters.

Step-4: Calculate the variance and place a new centroid of each cluster.

Step-5: Repeat the third steps, which means reassign each datapoint to the new closest centroid of each cluster.

Step-6: If any reassignment occurs, then go to step-4 else go to FINISH.

Step-7: The model is ready.

Program:

```
import numpy as np
import pandas as pd
# for visualizations
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
from datetime import datetime # Time Series analysis.
df = pd.read_csv(r"C:\Users\jaiwi\Desktop\Weather.csv")

df.head()

df = df.drop('Unnamed: 0', 1)

df

#making an attribute that would contain date (month, year). So that we
could get temperature values with the timeline.
df1 = pd.melt(df, id_vars='YEAR', value_vars=df.columns[1:]) ## melting
the data
df1.head()
```

```
df1['Date'] = df1['variable'] + ' ' + df1['YEAR'].astype(str)
df1.loc[:,'Date'] = df1['Date'].apply(lambda x : datetime.strptime(x, '%b
%Y')) ## Converting String to datetime object
df1.head()


df1.columns=['Year', 'Month', 'Temprature', 'Date']
df1.sort_values(by='Date', inplace=True) ## To get the time series right.
fig = go.Figure(layout = go.Layout(yaxis=dict(range=[0,
df1['Temprature'].max()+1])))
fig.add_trace(go.Scatter(x=df1['Date'], y=df1['Temprature']), )
fig.update_layout(title='Temprature Throught Timeline:',
                  xaxis_title='Time', yaxis_title='Temprature in Degrees')
fig.update_layout(xaxis=go.layout.XAxis(
    rangeselector=dict(
        buttons=list([dict(label="Whole View", step="all"),
                      dict(count=1,label="One Year
View",step="year",stepmode="todate")
                     ])),
        rangeslider=dict(visible=True),type="date")
)
fig.show()


#Insights:
#May 1921 has been the hottest month in india in history.
#Dec, Jan and Feb are the coldest months(“Winter”)
#Apr, May, Jun, July and Aug are the hottest months.(“Summer”).

fig = px.box(df1, 'Month', 'Temprature')
fig.update_layout(title='Warmest, Coldest and Median Monthly Tempratue.')
fig.show()

#July is the month with least Standard Deviation.
#So,temperature in July vary least. We can expect any day in July to be a
warm day.

from sklearn.cluster import KMeans
sse = []
target = df1['Temprature'].to_numpy().reshape(-1,1)
```

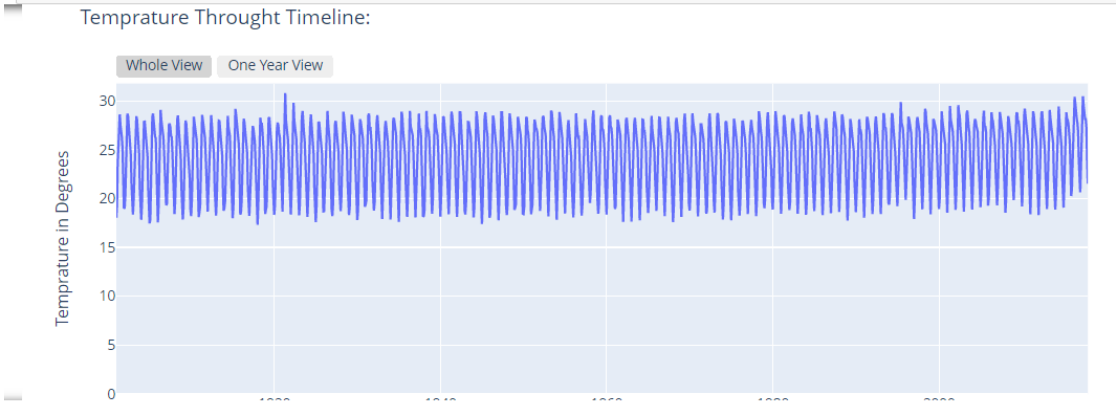
```
num_clusters = list(range(1, 10))

for k in num_clusters:
    km = KMeans(n_clusters=k)
    km.fit(target)
    sse.append(km.inertia_)

fig = go.Figure(data=[
    go.Scatter(x = num_clusters, y=sse, mode='lines'),
    go.Scatter(x = num_clusters, y=sse, mode='markers')
])

fig.update_layout(title="Evaluation on number of clusters:",
                  xaxis_title = "Number of Clusters:",
                  yaxis_title = "Sum of Squared Distance",
                  showlegend=False)
fig.show()

km = KMeans(3)
km.fit(df1['Temprature'].to_numpy().reshape(-1,1))
df1.loc[:,'Temp Labels'] = km.labels_
fig = px.scatter(df1, 'Date', 'Temprature', color='Temp Labels')
fig.update_layout(title = "Temprature clusters.",
                  xaxis_title="Date", yaxis_title="Temprature")
fig.show()

# we can see 3 main clusters based on temperatures.
# Jan, Feb and Dec are the coldest months.
# Apr, May, Jun, Jul, Aug and Sep; all have hotter temperatures.
# Mar, Oct and Nov are the months that have temperatures neither too hot
nor too cold.

fig = px.histogram(x=df1['Temprature'], nbins=200, histnorm='density')
fig.update_layout(title='Frequency chart of temprature readings:',
                  xaxis_title='Temprature', yaxis_title='Count')

# There is a cluster from 26.2-27.5 and mean temperature for most months
during history has been between 26.8-26.9
```

```
df['Yearly Mean'] = df.iloc[:,1:].mean(axis=1) ## Axis 1 for row wise and
axis 0 for columns.
fig = go.Figure(data=[
    go.Scatter(name='Yearly Tempratures' , x=df['YEAR'], y=df['Yearly
Mean'], mode='lines'),
    go.Scatter(name='Yearly Tempratures' , x=df['YEAR'], y=df['Yearly
Mean'], mode='markers')
])
fig.update_layout(title='Yearly Mean Temprature :',
                  xaxis_title='Time', yaxis_title='Temprature in Degrees')
fig.show()

fig = px.line(df1, 'Year', 'Temprature', facet_col='Month',
facet_col_wrap=4)
fig.update_layout(title='Monthly temprature throught history:')
fig.show()

df['Winter'] = df[['DEC', 'JAN', 'FEB']].mean(axis=1)
df['Summer'] = df[['MAR', 'APR', 'MAY']].mean(axis=1)
df['Monsoon'] = df[['JUN', 'JUL', 'AUG', 'SEP']].mean(axis=1)
df['Autumn'] = df[['OCT', 'NOV']].mean(axis=1)
seasonal_df = df[['YEAR', 'Winter', 'Summer', 'Monsoon', 'Autumn']]
seasonal_df = pd.melt(seasonal_df, id_vars='YEAR',
value_vars=seasonal_df.columns[1:])
seasonal_df.columns=['Year', 'Season', 'Temprature']

fig = px.scatter(seasonal_df, 'Year', 'Temprature', facet_col='Season',
facet_col_wrap=2, trendline='ols')
fig.update_layout(title='Seasonal mean tempratures throught years:')
fig.show()

px.scatter(df1, 'Month', 'Temprature', size='Temprature',
animation_frame='Year')

from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score

df2 = df1[['Year', 'Month', 'Temprature']].copy()
df2 = pd.get_dummies(df2)
```

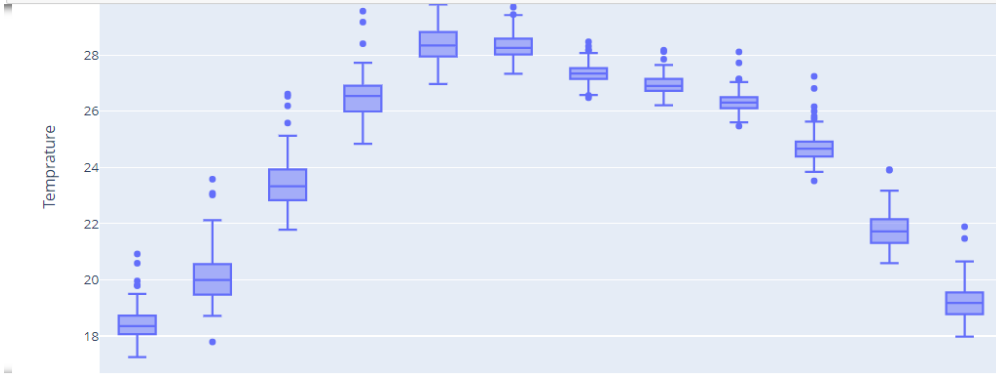
```
y = df2[['Temprature']]
x = df2.drop(columns='Temprature')

dtr = DecisionTreeRegressor()
train_x, test_x, train_y, test_y = train_test_split(x,y,test_size=0.3)
dtr.fit(train_x, train_y)
pred = dtr.predict(test_x)
r2_score(test_y, pred)

next_Year = df1[df1['Year']==2017][['Year', 'Month']]
next_Year.Year.replace(2017,2018, inplace=True)
next_Year= pd.get_dummies(next_Year)
temp_2018 = dtr.predict(next_Year)

temp_2018 = {'Month':df1['Month'].unique(), 'Temprature':temp_2018}
temp_2018=pd.DataFrame(temp_2018)
temp_2018['Year'] = 2018
temp_2018


forecasted_temp = pd.concat([df1,temp_2018],
sort=False).groupby(by='Year')['Temprature'].mean().reset_index()
fig = go.Figure(data=[
    go.Scatter(name='Yearly Mean Temprature', x=forecasted_temp['Year'],
y=forecasted_temp['Temprature'], mode='lines'),
    go.Scatter(name='Yearly Mean Temprature', x=forecasted_temp ['Year'],
y=forecasted_temp['Temprature'], mode='markers')
])
fig.update_layout(title='Forecasted Temprature:',
                  xaxis_title='Time', yaxis_title='Temprature in Degrees')
fig.show()
```

Output:

```
df1.columns=['Year', 'Month', 'Temprature', 'Date']
df1.sort_values(by='Date', inplace=True) ## To get the time series right.
fig = go.Figure(layout = go.Layout(yaxis=dict(range=[0, df1['Temprature'].max()+1])))
fig.add_trace(go.Scatter(x=df1['Date'], y=df1['Temprature']), )
fig.update_layout(title='Temprature Throught Timeline:',
                  xaxis_title='Time', yaxis_title='Temprature in Degrees')
fig.update_layout(xaxis=go.layout.XAxis(
    rangeselector=dict(
        buttons=list([dict(label="Whole View", step="all"),
                      dict(count=1,label="One Year View",step="year",stepmode="todate")
                     ])),
        rangeslider=dict(visible=True),type="date")
)
fig.show()
```

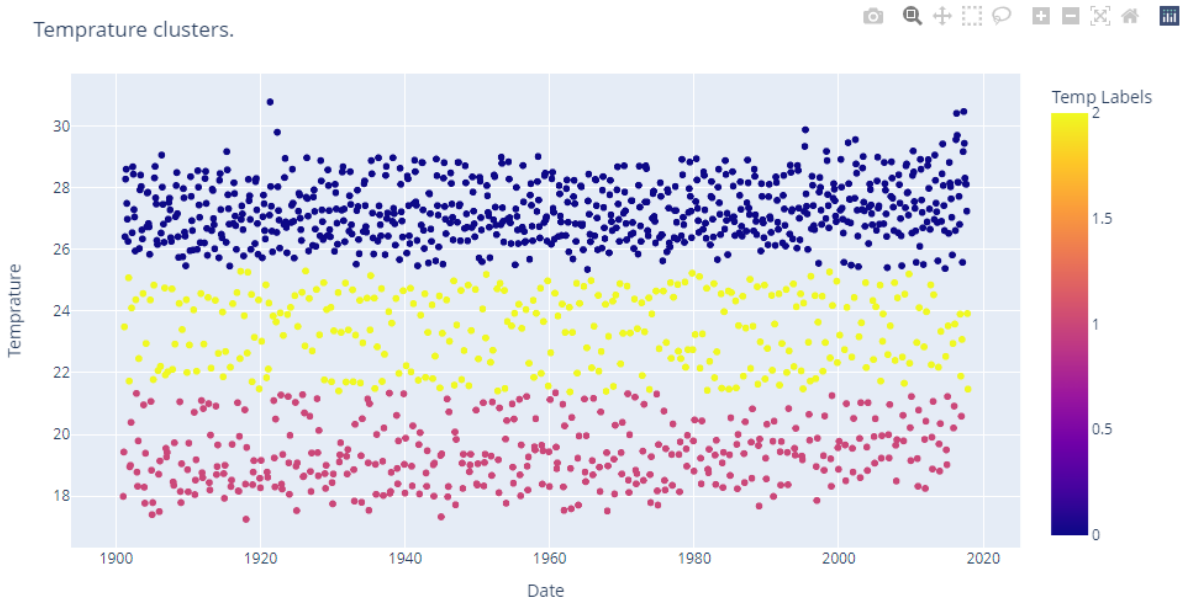



```
fig = px.box(df1, 'Month', 'Temprature')
fig.update_layout(title='Warmest, Coldest and Median Monthly Tempratue.')
fig.show()
```

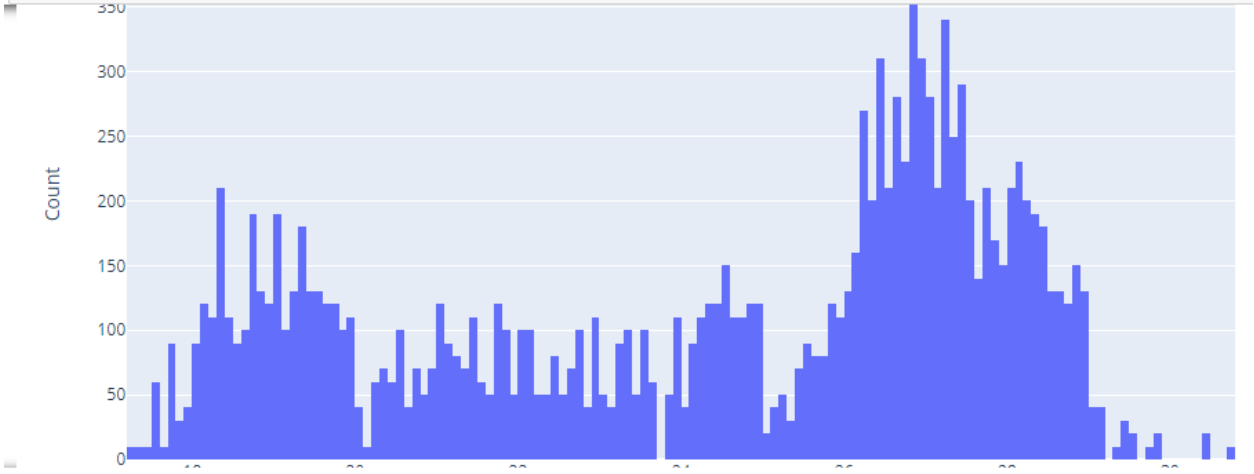


```
for k in num_clusters:
    km = KMeans(n_clusters=k)
    km.fit(target)
    sse.append(km.inertia_)

fig = go.Figure(data=[
    go.Scatter(x = num_clusters, y=sse, mode='lines'),
    go.Scatter(x = num_clusters, y=sse, mode='markers')
])

fig.update_layout(title="Evaluation on number of clusters:",
                  xaxis_title = "Number of Clusters:",
                  yaxis_title = "Sum of Squared Distance",
                  showlegend=False)
fig.show()
```

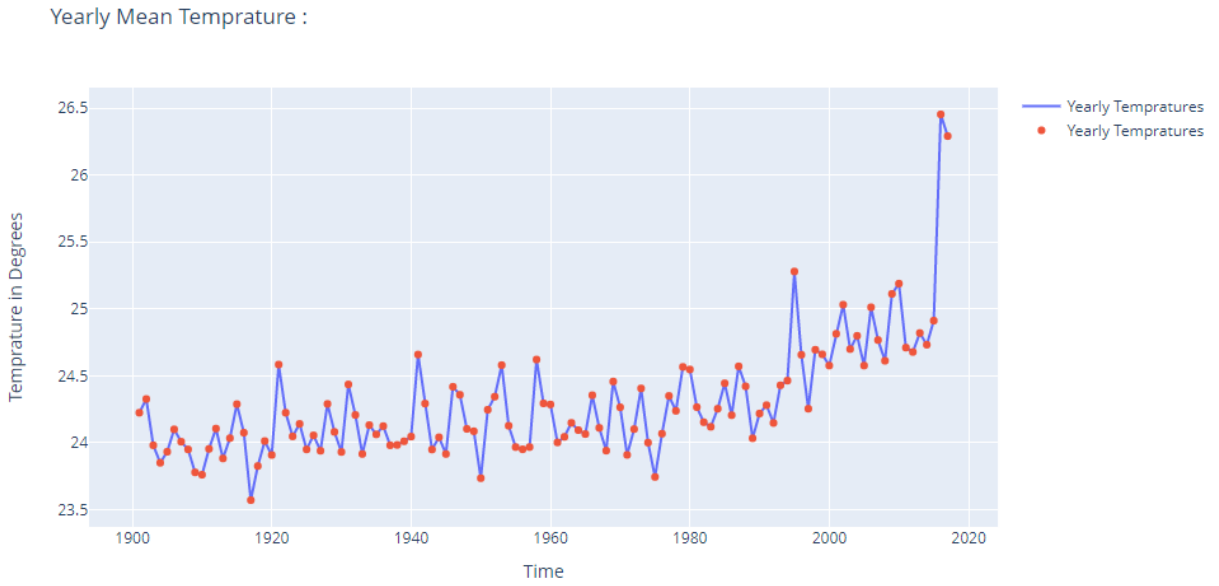
```
km = KMeans(3)
km.fit(df1['Temprature'].to_numpy().reshape(-1,1))
df1.loc[:,'Temp Labels'] = km.labels_
fig = px.scatter(df1, 'Date', 'Temprature', color='Temp Labels')
fig.update_layout(title = "Temprature clusters.",
                  xaxis_title="Date", yaxis_title="Temprature")
fig.show()
```



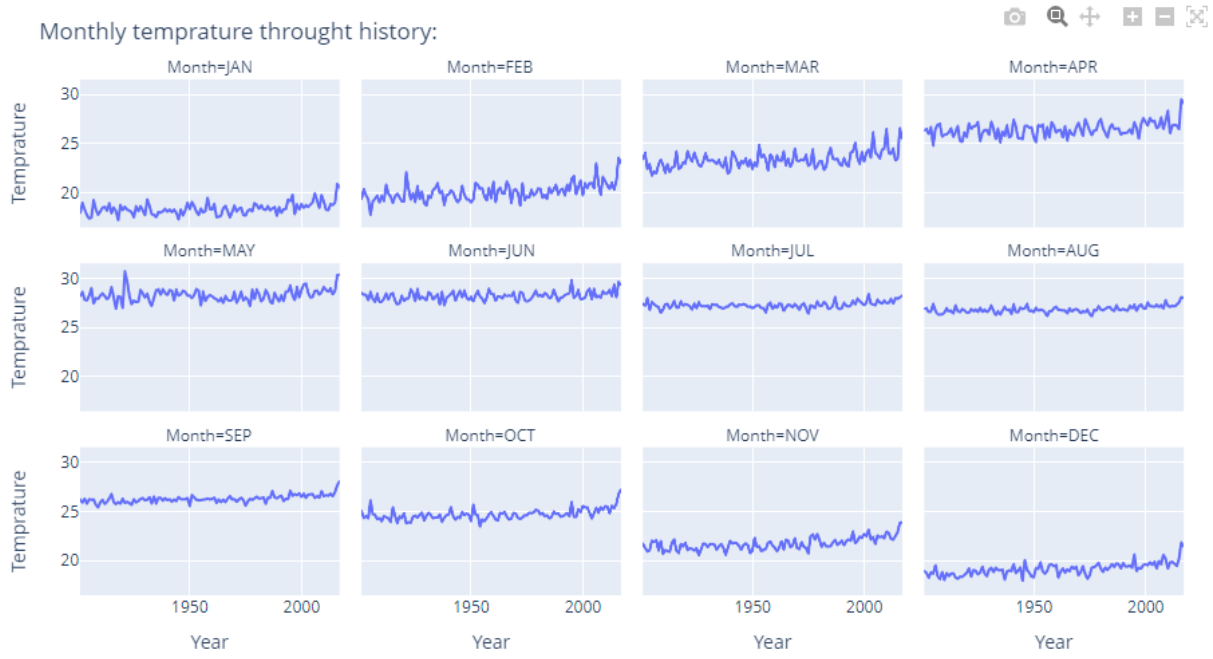

```
fig = px.histogram(x=df1['Temprature'], nbins=200, histnorm='density')
fig.update_layout(title='Frequency chart of temprature readings:',
                  xaxis_title='Temprature', yaxis_title='Count')
```



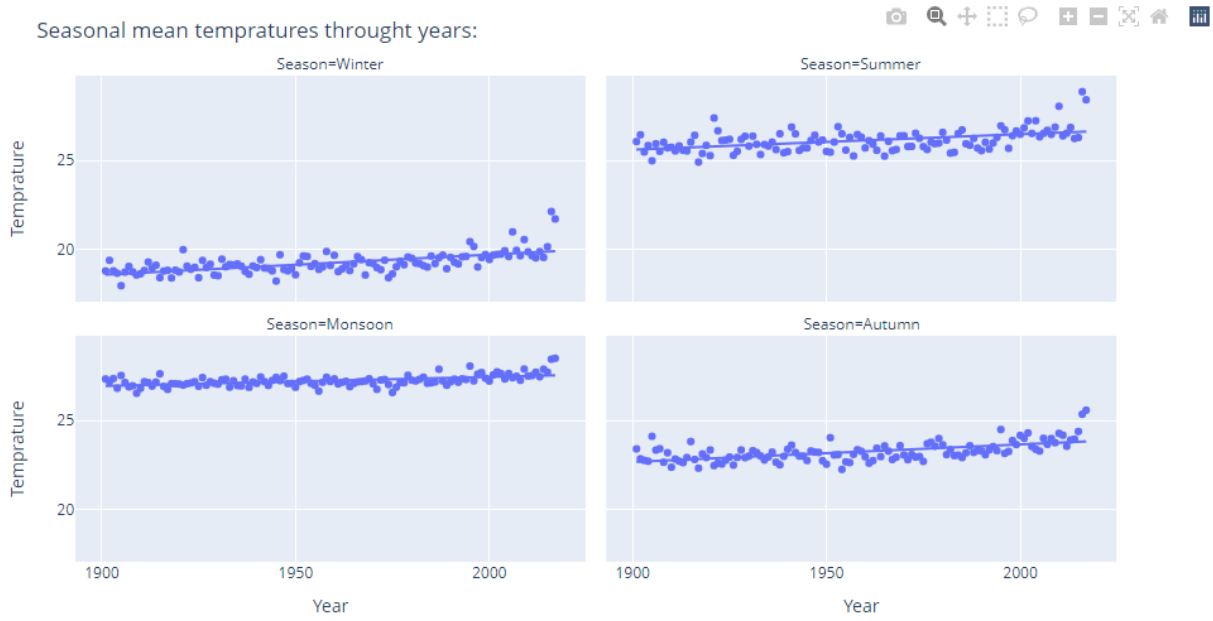
```
df['Yearly Mean'] = df.iloc[:,1:].mean(axis=1) ## Axis 1 for row wise and axis 0 for columns.
fig = go.Figure(data=[
    go.Scatter(name='Yearly Tempratures' , x=df['YEAR'], y=df['Yearly Mean'], mode='lines'),
    go.Scatter(name='Yearly Tempratures' , x=df['YEAR'], y=df['Yearly Mean'], mode='markers')
])
fig.update_layout(title='Yearly Mean Temprature :',
                  xaxis_title='Time', yaxis_title='Temprature in Degrees')
fig.show()
```



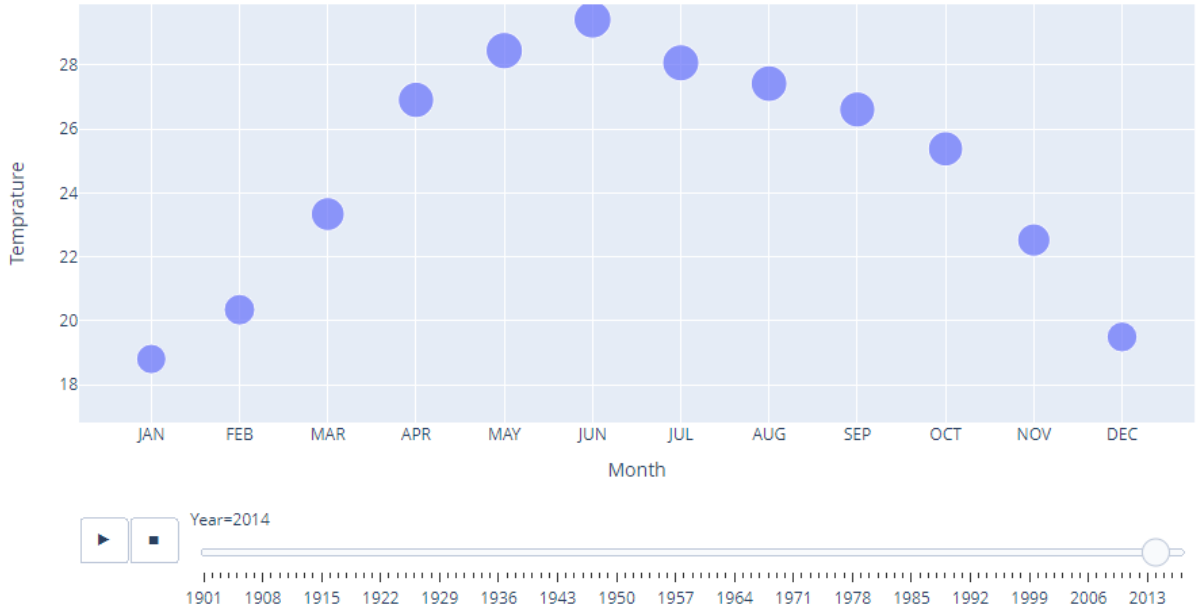

```
fig = px.line(df1, 'Year', 'Temprature', facet_col='Month', facet_col_wrap=4)
fig.update_layout(title='Monthly temprature throught history:')
fig.show()
```



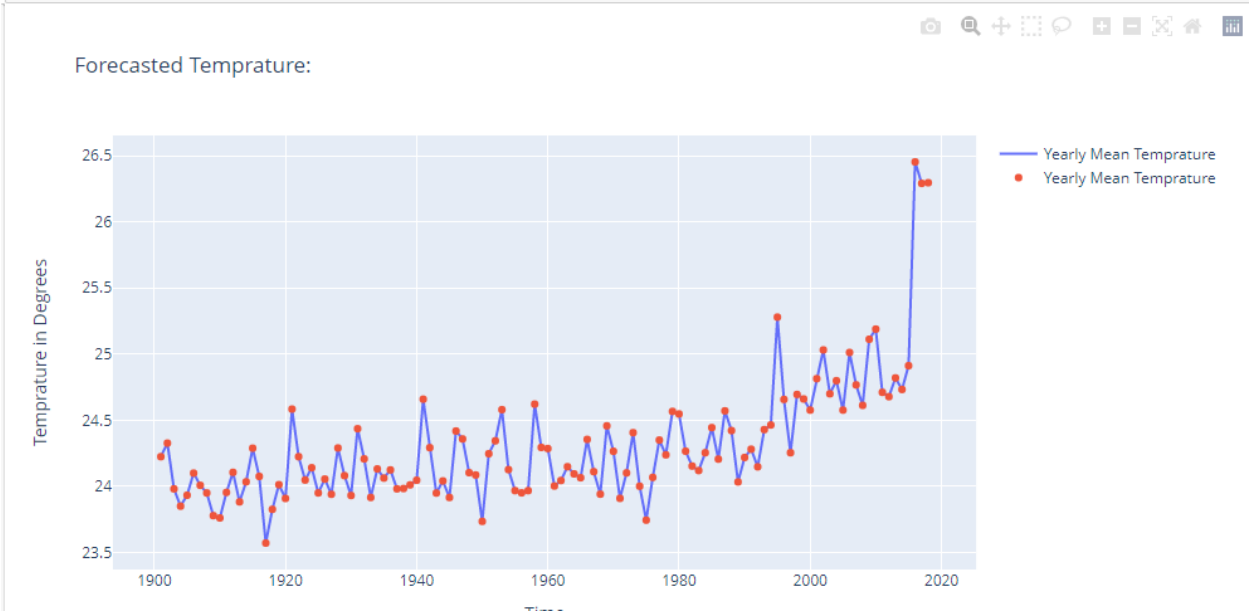
```
df['Winter'] = df[['DEC', 'JAN', 'FEB']].mean(axis=1)
df['Summer'] = df[['MAR', 'APR', 'MAY']].mean(axis=1)
df['Monsoon'] = df[['JUN', 'JUL', 'AUG', 'SEP']].mean(axis=1)
df['Autumn'] = df[['OCT', 'NOV']].mean(axis=1)
seasonal_df = df[['YEAR', 'Winter', 'Summer', 'Monsoon', 'Autumn']]
seasonal_df = pd.melt(seasonal_df, id_vars='YEAR', value_vars=seasonal_df.columns[1:])
seasonal_df.columns=['Year', 'Season', 'Temprature']

fig = px.scatter(seasonal_df, 'Year', 'Temprature', facet_col='Season', facet_col_wrap=2, trendline='ols')
fig.update_layout(title='Seasonal mean tempratures throught years:')
fig.show()
```




```
px.scatter(df1, 'Month', 'Temprature', size='Temprature', animation_frame='Year')
```



```
forecasted_temp = pd.concat([df1,temp_2018], sort=False).groupby(by='Year')['Temprature'].mean().reset_index()
fig = go.Figure(data=[
    go.Scatter(name='Yearly Mean Temprature', x=forecasted_temp['Year'], y=forecasted_temp['Temprature'], mode='lines'),
    go.Scatter(name='Yearly Mean Temprature', x=forecasted_temp ['Year'], y=forecasted_temp['Temprature'], mode='markers')
])
fig.update_layout(title='Forecasted Temprature:',
                  xaxis_title='Time', yaxis_title='Temprature in Degrees')
fig.show()
```



Conclusion:

In this mini project we have created a weather forecasting program that predicts the Temperature.

In this project ,we are using a dataset that includes the monthly temperature of each year starting from 1901 till 2017. We did the initial data cleaning to make sure the data doesn't remain skewed and only the required rows and columns were included. To understand the data in a better way, we are using the plotly library in python to give a better visualization of the same. Using the same, we understand the median temperature of every month all these years. Plotting the same makes the job easier. Using the standard deviation feature, we understood that july had the least Standard Deviation. So,the temperature in July varies the least. Next we used K means clustering to understand the distribution of the data according to the average temperature. This gave us a pretty good idea about how the temperature varies. So, we understood that we can see 3 main clusters based on temperatures.Jan, Feb and Dec were the coldest months. Apr, May,

Jun, Jul, Aug and Sep all have hotter temperatures. Mar, Oct and Nov are the months that have temperatures neither too hot nor too cold. Next,to understand the mean temperature of the year, we have used a histogram. We understand that after 2015, yearly temperature has increased drastically. We saw a monthly like up-down pattern in yearly temperature. We found that after 2015, global warming increased and this led to increase in the mean temperature. We saw a positive trend line between temperature and time. And at last, we used a decision tree to check our prediction. The r2 score achieved was significant enough to help us back our assumptions about climate change.